Полная логика проекта — пошагово

Ниже — Описано поведение при старте, калибровках, всех режимах (ручном и Peck/PCB), меню, управлении вводом и безопасностью.

0. Обозначения

- Координатная система: **Z = 0** — нижний концевик (Z-min).
- `posLimitTop` — максимально допустимое Z (верхняя граница хода), вычисляется как `posMax - 12.0` мм (после калибровки концевиков).
- `surface_mm` — позиция поверхности материала (мм) в той же системе координат (то есть положение поверхности относительно Z=0).
- `posDrillPCBStart = surface_mm + 2.0` мм (всегда 2 мм **выше** поверхности). Это обязательное правило.
- Все концевики NC → GND (в коде читаем `LOW` как «сработал»). Touch Plate тоже активный LOW.
- Шаги/мм: `stepsPerMM = MOTOR_STEPS * microsteps / SCREW_LEAD`. *Примеры*: MOTOR_STEPS=200, SCREW_LEAD=4 мм/об, microsteps=16 → stepsPerMM = 200*16/4 = 800 шага/мм.
- Скорости: пользователь вводит в мм/мин → переводим в шаги/сек для stepper: `stepsPerSec = (mmPerMin / 60) * stepsPerMM`.

1. Поведение при старте (power-on / reset)

1. Инициализация:
    - инициализировать I²C (Wire), дисплей (PCF8574), энкодер (прерывания A и кнопка), все входы/выходы (концевики, кнопки), PWM-таймеры (PA8, PA9).
    - конфигурировать STEP/DIR/EN/MS1/MS2 как выходы.
2. Попытка загрузить конфиг из EEPROM (24LC256) и проверить CRC:
    - если конфиг валиден → применить: microsteps (установить MS1/MS2), вычислить `stepsPerMM`, настроить PWM и LED;
    - если нет → загрузить значения по умолчанию и записать в EEPROM.
3. Немедленно выполнить **калибровку концевиков** (обязательная):
    - опускание вниз до Z-min → установить позицию 0;
    - подъём до Z-max → записать `posMax`;
    - `posLimitTop = max(0, posMax - 12.0);`
    - установить `calibratedZ = true` и сохранить.
      (Если по какой-то причине концевики не срабатывают — уйти в STATE_EMERGENCY и вывести ошибку.)
4. Если `calibratedPlate == false`, на главный экран выводится приглашение выполнить калибровку Touch Plate (предложение, не обязательное — но без обеих калибровок работа сверления блокируется по безопасности).

5. Показать главный экран — режим Idle.

2. Главные состояния

- `STATE_IDLE` — ожидание ввода; отображение статуса.
- `STATE_IN_MENU` — пользователь изменяет параметры; движение заблокировано.
- `STATE_CALIBRATING_Z` — автоматическая калибровка по концевикам.
- `STATE_PROBING_PLATE` — калибровка по Touch Plate.
- `STATE_MANUAL_DRILL` — ручное сверление (Drill button удерживается).
- `STATE_PECK_DRILL` — автоматический Peck Drill (Drill PCB).
- `STATE_EMERGENCY` — аварийный стоп (концевик/ошибка/прерывание).

Переключение между состояниями только по командам/условиям, при входе/выходе — всегда сохранять/восстанавливать PWM и стоп шаговика.

3. Калибровка по концевикам (Z)

1. Переход в `STATE_CALIBRATING_Z`.
2. Медленно опуститься вниз (small feed) до срабатывания `Z-min` (LOW).
3. Установить текущую позицию шагов в 0 (`stepper.setCurrentPosition(0)`).
4. Подняться вверх до срабатывания `Z-max`. Записать `posMax = stepsToMM(currentSteps)`.
5. `posLimitTop = max(0, posMax - 12.0)`.
6. Пометить `calibratedZ = true`, записать в EEPROM.
7. Выход в `STATE_IDLE`.

Защита: при любом движении проверять концевики; при неожиданном срабатывании — emergencyStop.

4. Калибровка Touch Plate (высота инструмента)

(Пункт меню / процедура)

1. Требование: `calibratedZ == true`. Если нет — предложить сначала калибровать Z.
2. Пользователь кладёт Touch Plate на заготовку и подтверждает (кнопка энкодера).
3. Подъехать в безопасную позицию сверху
4. Медленное опускание (низкая скорость) до срабатывания `TOUCH` (LOW).
5. Когда Touch Plate сработал — фиксируем `measured_mm = stepsToMM(currentSteps)`.
6. `surface_mm = measured_mm - plateThickness` (plateThickness задаётся в меню).
    - Ограничить: `surface_mm = clamp(surface_mm, 0, posLimitTop)`.

7. Автоматически: `posDrillPCBStart = surface_mm + 2.0` (2 мм выше поверхности). Обязательно сохранить в EEPROM.
8. `calibratedPlate = true`. Вывести результат на LCD. Выход в `STATE_IDLE`.

Примечание: Touch Plate должен быть изолирован и иметь контакт с инструментом (инструмент → GND).

5. Пункт 3.3 «Проверка поверхности»

- Опция в меню. Поведение:
    1. Проверка флагов `calibratedZ` и `calibratedPlate`. Если нет — ошибка.
    2. По подтверждению: подъехать к `surface_mm` (медленно).
    3. Остановиться и показать `Z = surface_mm` на LCD.
    4. При нажатии — подъехать к `posDrillPCBStart` (surface + 2 mm) и показать позицию.
    5. Возврат к `posDrillStart` после проверки.
- Это функциональное тестирование, чтобы визуально убедиться в корректности калибровки.

6. Ручное сверление (Drill)

- Кнопка Drill — активна при удержании (нормально HIGH, замкнута на GND при нажатии).
- Алгоритм:
    1. Проверить флаги калибровок; если отсутствуют — предупредить и не начинать.
    2. Включить шпиндель PWM на `speedPWMDrill` %.
    3. Пока кнопка удерживается:
        - двигать вниз с шагом/итерацией (например, по 0.1–0.2 мм) со скоростью `speedDrillDrill` (мм/мин), каждая итерация — проверка границ:
            - не опускаться ниже `(surface_mm - maxManualDepth)` (защита по глубине);
            - не заходить ниже 0 (Z >= 0);
            - контролировать Z-min (если сработал — emergencyStop).
    4. По отпусканию кнопки: выключить шпиндель и возврат в `posDrillStart` по `speedMoveDrill`.
- Дополнительно: при удержании кнопки оператор вручную ограничивает глубину → system respects `maxManualDepth`.

7. Peck Drill — Drill PCB (автомат)

- Цель: сверлить до `drill_PCBDepth` в ступенях `peckDepth`, возвращаясь после каждого захода в `posDrillPCBStart`.

- Алгоритм:
    1. Проверить калибровки.
    2. Перейти в `posDrillPCBStart` (движение с `speedMovePeck`).
    3. `currentDepth = 0`
    4. Пока `currentDepth < drill_PCBDepth`:
        - `dive = min(peckDepth, drill_PCBDepth - currentDepth)`.
        - `target_mm = surface_mm - (currentDepth + dive)` — **пояснение:** уменьшаем Z (вниз) на глубину.
        - `target_mm = clamp(target_mm, 0.0, posLimitTop)` — защита.
        - Включить шпиндель на `speedPWMDrillPCB`.
        - Перемещение вниз к `target_mm` со скоростью `speedDrillPeck`.
        - Остановить шпиндель.
        - Подняться обратно в `posDrillPCBStart` со скоростью `speedMovePeck`.
        - `currentDepth += dive`.
    5. По завершении отключить шпиндель, вернуться в `posDrillPCBStart` и `STATE_IDLE`.
- Защиты во время каждого движения:
    1. Контроль Z-min/Z-max.
    2. Контроль posLimitTop и нижней границы (0).

8. Смена инструмента (Tool Change)

- Кнопка ToolChange:
    1. При нажатии → подняться вверх до `Z-max` (верхний концевик), установить позицию `posLimitTop` (или просто убедиться в нахождении у верхнего лимита).
    2. Остановиться; затем опуститься **на 3 оборота ШВП (3 * SCREW_LEAD = 12 мм)** вниз и стоп.
    3. Это позиция для смены инструмента; шпиндель выключён.
    4. Возврат operator-initiated (по кнопке или автоматически в posDrillStart).
- Защита: мониторить концевики; если Z-min сработал во время подъёма — emergencyStop.

---

9. Меню и параметры (структура + поведение)

- Структура меню (главная ветка):

1. Режимы сверления:
    - Pos Drill Start (редактируемый)
    - Pos Drill PCB (авто; показывается)
    - Peck Depth
    - PCB Total Depth
    - Speed Move Drill
    - Speed Drill Drill
    - Speed Move Peck
    - Speed Drill Peck
    - PWM Spindle Drill (%)
    - PWM Spindle PCB (%)
2. Настройки оборудования:
    - Microsteps (1 / 2 / 4 / 8 / 16) — изменение устанавливает MS1/MS2 и пересчитывает stepsPerMM
    - Plate thickness
    - LED Brightness
3. Калибровка:
    - Calibrate Z
    - Calibrate Touch Plate
    - Check Surface (пункт 3.3)
4. Сохранить конфиг
5. Exit

- Поведение редактора:
    1. Энкодер вращает значение (delta → ±N * step).
    2. Кнопка энкодера подтверждает (короткое) или входит/выходит (долгое).
    3. При изменении microsteps — установить пины MS1/MS2 (учесть LL-тоггл поведение), пересчитать `stepsPerMM`, пересчитать ограничения.
    4. При изменении `surface_mm` вручную — автоматически скорректировать `posDrillPCBStart = surface_mm + 2.0`; сохранить по запросу.

10. EEPROM — хранение конфигурации

- В EEPROM (24LC256) сохраняем структуру `Config`, включающую все параметры, флаги калибровок и `posLimitTop`/`posMax`.
- Перед записью вычисляем CRC16-CCITT и записываем в конец блока. При загрузке — проверяем CRC.
- Формат: бинарный блок `sizeof(Config)` (фиксированная длина).
- Запись: блоками по 64 байта (page write) с ожиданием tWR (≈5–10 ms).

- При изменении критичных параметров (microsteps, plateThickness, surface_mm, speeds) — сохранять по явному запросу (Save) и/или при выходе из меню, а иногда автоматически (encoder short press = quick save).

11. Управление микрошагами TMC2209 (MS1/MS2)

- Управление аппаратными пинами MS1/MS2; учесть особенность драйвера:
  - LL → 1/1 ↔ 1/16 (повторный LL переключает между 1/1 и 1/16).
  - HL → 1/2 ↔ 1/32
  - LH → 1/4 ↔ 1/64
  - HH → 1/8 ↔ 1/128
- При старте прошивки: прочитать `cfg.microsteps` из EEPROM и установить MS1/MS2 (выполнить нужную последовательность, возможно двойной LL).
- В меню «Microsteps» — менять значение (1/2/4/8/16) и применять сразу:
  - установить пины MS1/MS2 корректно,
  - вызвать `computeStepsPerMM()` и пересчитать скорости/параметры зависимости.
- Защита: запретить редкие значения, обязать значение из списка {1,2,4,8,16}.

12. Управление PWM (шпиндель и LED)

- Использовать аппаратный таймер (TIM1 CH1 → PA8 и CH2 → PA9) настроенный на ~60 kHz (или 30kHz для шпинделя по желанию).
- PWM задаётся в % (0–100). В меню для шпинделя — два параметра: `speedPWMDrill` и `speedPWMDrillPCB`.
- Для LED (DF6113 DIM) соблюдаем ограничение 10–100% (DF6113 не корректно реагирует на очень малые скважности).
- Изменение значения в меню — немедленное применение (applyImmediately=true при редактировании).
- При аварии/останове — PWM = 0.

13. Работа с энкодером и кнопками (ввод)

- **Энкодер A** — прерывание CHANGE; в ISR читаем B и корректно приращиваем/уменьшаем `encoderPos` (очень коротко, минимум работы); не вызывать LCD/Stepper из ISR.
- **Энкодер кнопка** — прерывание FALLING (установить флаг `encoderBtnPressed = true`), но реальные действия (дебаунс/определение долгого/короткого нажатия) выполнять в основном цикле.
- **Остальные кнопки (Drill, DrillPCB, UP, DOWN, ToolChange)** — использовать Bounce2 или поллинг+дебаунс в `loop()`; можно использовать `fell()/rose()` для событий.
- Правило ISR: никаких длительных операций (delay, I2C, stepper.run) — только

атомарные флаги/счётчики.

14. Проверки безопасности и аварии

- В любой функции движения (moveToPositionMM / moveRelativeMM / stepper.run loop) — постоянно проверять:
    - `if (digitalRead(Z_MIN) == LOW && movingDown) emergencyStop("Z-min")`
    - `if (digitalRead(Z_MAX) == LOW && movingUp) emergencyStop("Z-max")`
    - Ограниечение по `posLimitTop` и по нижней границе 0.
- EmergencyStop:
    - Остановить stepper, установить PWM шпинделя = 0, пометить состояние `STATE_EMERGENCY`, вывести сообщение на LCD, ждать ручного вмешательства (сброс флага/перезапуск).
    - В логе/Serial можно вывести причину для отладки.
- Защита параметров: при установке глубин/скоростей — clamp значений к разумным пределам.

15. Преобразования и формулы

- `stepsPerMM = MOTOR_STEPS * microsteps / SCREW_LEAD`
- `stepsTarget = round(mm * stepsPerMM)`
- `mmCurrent = stepsCurrent / stepsPerMM`
- `stepsPerSec = (mmPerMin / 60.0) * stepsPerMM`
- `AccelStepper` принимает `setMaxSpeed()` в шагах/сек и `setAcceleration()` в шагах/сек².

16. Сохранение макросов / G-code (если потребуется)

- Текущая логика генерирует G-подобные последовательности динамически (peck routine) на MCU, хранит только параметры (позиции, глубины, скорости).
- Если нужен функционал «сохранить N макросов с разными параметрами» — выделить область EEPROM для массива записей `{peckDepth, drillDepth, speedMove, speedDrill, posStart}` и индекс/имена; при выборе — загружать и выполнять. (Примечание: это расширение — делается по запросу.)

17. Протоколы логирования и отладки

- Серийный порт (Serial 115200) — логирование старт/ошибок/событий (калибровка, emergency, EEPROM load/save).
- LCD показывает краткую информацию; для подробного лога — использовать Serial.

18. Взаимодействие задач

- **ISR**: только флаги/счётчики; все тяжёлые действия в `loop()` или задачах.
- Движения — последовательны: одна операция движения (moveToPositionMM / performPeckDrillRoutine) блокирует другие пользовательские команды (используем состояние) — кнопки должны задавать запросы, а не выполнять движение прямо.
- При экстренной ситуации — флаги прерывают цикл движения и вызывают emergencyStop.

19. Поведение при некорректных параметрах

- При вводе параметра, который физически выводит инструмент за пределы (например, `posDrillStart > posLimitTop`), GUI должен не позволить сохранить или автоматически ограничить его.
- Перед стартом Peck/Drill — вычисляется контрольный маршрут; если любой target выходит за 0..posLimitTop — блокируется с сообщением.

20. контрольный список логики

1. Старт: init HW → load EEPROM (CRC) → set microsteps → compute stepsPerMM → mandatory calibrate Z → ask/perform plate probe.
2. Все движения проверяют Z-min/Z-max и posLimitTop/0.
3. Touch Plate калибровка автоматически корректирует posDrillPCBStart = surface + 2 mm.
4. Manual Drill: шпиндель ON при удержании, движение вниз шагами, не глубже `maxManualDepth`, возврат в posDrillStart.
5. Peck Drill: автоматический цикл dive/raise до общей глубины, с PWM control и подъемом между заходами.
6. Menu: редактирование всех параметров, microsteps устанавливаются через MS1/MS2 с учетом особенностей LL комбинации.
7. EEPROM: хранение конфигурации + CRC16, запись pagewise, чтение при старте.
8. Энкодер — прерывания; кнопки — debounced polling; ISR максимально «лёгкие».
9. EmergencyStop — немедленный стоп шаговика и PWM, вывод ошибки, блокировка операций до ручного сброса.